"EXPRESS MAIL" Mailing Label No.
EL840960609 US
Date of Deposit: March 30, 2001

INTERRUPT THROTTLING FOR INTER-PROCESSOR COMMUNICATIONS

This application claims the priority under 35 U.S.C. 119(e)(1) of copending U.S. provisional application number 60/194,258 filed on April 3, 2000, incorporated herein by reference.

FIELD OF THE INVENTION

The invention relates generally to data processing systems and, more particularly, to communications between data processors in a data processing system.

BACKGROUND OF THE INVENTION

In conventional data processing systems including a plurality of data processors which communicate with one another, such communication is often controlled by an interrupt mechanism. For example, if a first data processor wishes to communicate with a second data processor, the first data processor applies to the second data processor an interrupt signal which interrupts the second data processor. Conversely, if the second data processor wishes to communicate with the first data processor, then the second data processor applies to the first data processor an interrupt signal which interrupts the

second data processor. Each of the data processors typically includes an interrupt service routine which then handles the requested communication.

Execution of the interrupt service routines disadvantageously adds overhead processing to the processing loads of the data processors. In addition, servicing an interrupt can be particularly disadvantageous if the interrupted data processor has a pipelined data processing architecture.

It is therefore desirable to reduce the amount of interrupt activity involved in controlling communications between data processors.

According to the invention, a first data processor will interrupt a second data processor for communication therewith only if the first data processor determines that I/O is blocked on the second data processor. By using the indication of whether or not I/O is blocked on the second data processor, the first data processor can advantageously avoid interrupting the second data processor unnecessarily.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 diagrammatically illustrates exemplary embodiments of a data processing system according to the invention.

FIGURE 2 is similar to FIGURE 1, and includes a conceptual illustration of a data communication channel between the data processors of FIGURE 1.

FIGURE 3 illustrates in more detail the use and control of the shared memory of FIGURE 1 according to the invention.

FIGURE 4 illustrates exemplary operations which can be performed by the embodiments of FIGURES 1-3.

DETAILED DESCRIPTION

FIGURE 1 diagrammatically illustrates exemplary embodiments of a data processing system according to the invention. The exemplary system of FIGURE 1 includes first and second data processors 11 and 13 coupled for communication with one another via respective communication ports 25 and 26, a bus 15 and a shared memory resource 17. In one exemplary embodiment, the data processor 11 can be a general purpose microprocessor (e.g. x86 or ARM), and the data processor 13 can be a special purpose data processor such as a digital signal processor (DSP). The data processors in FIGURE 1 utilize respective multi-tasking operating systems that are capable of inter-processor signaling, such as generation and handling of inter-processor hardware interrupts. Access to the shared memory 17 can be arbitrated sufficiently to provide each processor with exclusive access to the shared memory, as described in further detail below.

As shown in FIGURE 1, the operating systems of the respective data processors 11 and 13 each include a multi-tasking kernel, memory management, hardware interrupts and device driver support. Each processor also includes an inter-processor communication (IPC) device driver. The IPC device driver 12 of the data processor 11 implements an interrupt throttling mechanism 16, and the IPC device driver 14 of the data processor 13 implements an interrupt throttling mechanism 18. The IPC device driver of each data processor uses its associated interrupt throttling mechanism to limit, or throttle, generation of IPC-related interrupts to the other data processor. By operation

of the interrupt throttling mechanisms, the data processing overhead incurred by the IPC device drivers for handling incoming interrupts is reduced.

FIGURE 1 further illustrates an application running on each of the data processors. These applications 20 and 22 communicate with one another via a communication channel data path as illustrated in FIGURE 2. The communication channel between the applications is illustrated conceptually within the broken lines of FIGURE 2. The actual physical data path passes through the shared memory 17 as illustrated. Referring again to the example of FIGURE 1, a man/machine interface (MMI) 19 is coupled to the data processor 11 for permitting a user to communicate with the application currently running on the data processor 11. As shown in FIGURE 2, a task A is associated with the application 20 on data processor 11, and a task B is associated with the application 22 on data processor 13. A task is a thread of execution that can be in various states in accordance with the associated data processor's operating system. The operating system kernel of each data processor has the ability to block, i.e. suspend, thread execution in the IPC device driver of that data processor. Also, the IPC device driver of each data processor has the ability to be signaled or interrupted by the other data processor, and each IPC device driver has access to the shared memory 17.

FIGURE 3 diagrammatically illustrates exemplary interfaces between the shared memory 17 and data processors 11 and 13. As shown in FIGURE 3, each of the data processors 11 and 13 can acquire exclusive access to task state attributes stored in the shared memory 17. In the example of FIGURE 3, such exclusive access is accomplished

by appropriately controlling a memory access apparatus 31, illustrated in FIGURE 3 as a data switch 31. For example, if the data processor 11 wishes to obtain exclusive access to the task state attributes stored in memory 17, the data processor 11 provides, via its memory interface, control signaling at 32 to cause the data switch 31 to assume the position illustrated in FIGURE 3. With the data switch in this configuration, the data processor 11 can access the shared memory 17 via the data path 34, and the data processor 13 is prevented from accessing the shared memory 17. Similarly, when the data processor 13 wishes to acquire exclusive access to the task state attributes in the shared memory 17, the memory interface of data processor 13 outputs appropriate control signaling at 33 to cause the data switch 31 to disconnect the data path 34 from the shared memory 17 and connect the data path 35 to the shared memory 17.

As illustrated in FIGURE 3, the shared memory 17 includes task I/O state attributes corresponding to task A on processor 11 and task B on processor 13. The task I/O state attribute of a given task does not necessarily represent the actual instantaneous operating system state of the task, but does indicate at least that the task is immediately going to block on, for example channel #1, or is immediately going to be unblocked with respect to channel #1 by an IPC interrupt. (This will become apparent from the description of FIGURE 4 hereinbelow.) The IPC device driver of each data processor can access the channel I/O attributes. Each IPC device driver includes interrupt throttle logic to implement its interrupt throttling mechanism.

Also shown in FIGURE 3 are IPC interrupt handlers in the IPC device drivers 12 and 14, and IPC hardware interrupt support portions in the operating systems of the data processors 11 and 13. Each data processor can thus interrupt and be interrupted by the other data processor, as described in more detail below.

FIGURE 4 illustrates exemplary operations which can be performed by either of the data processors 11 and 13 of FIGURES 1-3 when one of tasks A and B (see also FIGURES 2 and 3) wishes to communicate with the other of tasks A and B. A given application task will request I/O to send data to or receive data from a cooperating task on the other (cooperating) data processor. At 42, buffers are either passed into the device driver or removed from a device queue. If it is determined at 43 that buffer attributes from the cooperating task on the other data processor exist in the shared memory 17, buffer data and attributes can be exchanged at 44 in order to execute a data exchange. This can be accomplished, for example, by swapping buffer data and attributes, or by copying buffer data and attributes, both of which are well known conventional procedures for implementing inter-processor communications through a shared memory resource.

After having acquired exclusive access to the task I/O state attributes in the shared memory 17 (see 45A), the task I/O state of the other data processor is determined at 45. If the task I/O state attribute in the shared memory indicates that the cooperating data processor is blocked, then the task I/O state attribute for the cooperating task is set to 0 (unblocked) in shared memory at 46, after which a hardware interrupt to the cooperating

data processor is generated at 47. If the task I/O state attribute at 45 indicates that the cooperating task is not blocked, or after generating the hardware interrupt at 47, it is determined at 48 whether more buffers exist. If so, then operations return to block 43. When it is determined at 48 that no more buffers exist (buffer exchange can continue until either processor is out of data buffers), the data processor sets its own task I/O state attribute to a value of 1 (blocked) at 49, thereby indicating that its task is blocked. Thereafter at 50, the operating system scheduler of the multi-tasking kernel is called to block the task. Note also that the data processor releases its exclusive shared memory access either after setting its own task I/O state to a value of 1 at 49 (see 49A) or after determining that more buffers exist at 48 (see 48A).

FIGURE 4 also indicates how the device driver responds to an IPC interrupt received from the cooperating processor. As illustrated in FIGURE 4, the IPC interrupt handler in the IPC device driver receives the IPC interrupt at 51, after which the operating system scheduler is called at 50 to unblock the task in response to the IPC interrupt. Operations proceed to block 48 after the scheduler blocks or unblocks the task at 50.

As illustrated in FIGURES 1-4, if a task running on the data processor 11, for example, wishes to communicate with a cooperating task running on the data processor 13, the data processor 11 will generate a hardware interrupt to the data processor 13 only if the data processor 11 determines that the I/O state of the cooperating task on data processor 13 is (or is about to be) blocked. In this manner, the data processor 13 is

interrupted only when necessary, i.e. only when the cooperating task is blocked. This avoids unnecessary interruption of the data processor 13, thereby advantageously reducing the interrupt handling overhead on data processor 13.

Although the exemplary embodiments of FIGURES 1-3 illustrate only two data processors, it will be apparent to workers in the art that the interrupt throttling mechanism according to the invention is applicable to communications between any two data processors. Accordingly, the interrupt throttling mechanism of, for example, data processor 11 can be utilized in conjunction with communications between data processor 11 and other data processors (not explicitly shown) which can also gain exclusive access to the shared memory 17. In some exemplary embodiments of the invention, all components (except MMI 19) of FIGURES 1-3 can be embedded together in a single integrated circuit and, in other embodiments, one or more of these components can be provided on an integrated circuit separately from the remaining components.

Referring again to FIGURE 1, the man/machine interface (MMI) 19 permits a user to communicate with the application 20. Examples of the man/machine interface 19 include a keyboard/keypad, a visual display, etc. Examples of the system of FIGURES 1-3 include a cellular telephone, a laptop computer and a set-top box.

It will also be evident to workers in the art that the systems of FIGURES 1-3 could also be implemented with the interrupt throttling mechanism provided on only one of the data processors. For example, it is possible to throttle interrupts to only one of the data processors, such as the data processor 13. This could be advantageous, for example,

if the system hardware limits the ability to provide exclusive access to the shared memory. If an exclusive access mechanism such as shown at 31 in FIGURE 3 is not available, then the data processor 11 could use, for example, a read-modify-write (RMW) bus cycle to atomically "test-and-set" the respective I/O state attributes of the two tasks.

Although exemplary embodiments of the invention are described above in detail, this does not limit the scope of the invention, which can be practiced in a variety of embodiments.